Comp. & Maths. with Appls. Vol. 1, pp. 327–336. Pergamon Press, 1975. Printed in Great Britain.

A STRUCTURED COMPUTER SYSTEM MODEL

ELÖD KNUTH

Computer and Automation Institute, Hungarian Academy of Sciences, H 1111, Budapest, Kende u. 13–17, Hungary

Communicated by E. Y. Rodin*

(*Received March* 1975; *in revised form, May* 1975)

Abstract—This paper describes a general mathematical model (mainly stochastic, in nature), for the basic operational structure of a multiprogrammed computer system. The model considered depends, on the one hand, on the job-flow and, on the other, upon the hardware configuration and operating system of the computer. A formal description is presented to illustrate the most important activities, in the language SIMULA 67. Using this model it is possible to give an exact form to the statistical problems of job mixing and to the non linear stochastic control problems of priorities and page fault problems.

1. INTRODUCTION

In the use of multiprogrammed computer systems it becomes obvious that the proper management of the job-flow is a prime necessity for efficient utilization. This depends first of all on the monitor (executive) of the operating system. The monitor has several strategies to allocate and to schedule the resources for the different demands, appearing at the same time.

Some of these activities are well studied in simple cases, but most of them are not sufficiently investigated.

In our earlier papers (see e.g. [1], [11]) we proposed a stochastic description of a multiprogrammed computer system together with the job-flow. Here we explain this in a more detailed form. The problems of such a description arose with respect to the parameter choices in the operating system of a CDC 3300, located in the Computing Center of the Hungarian Academy of Sciences. On the other hand we also want to give a more or less adequate mathematical model of such interesting experimental research as described in the papers of Marshall[4] and Denning[7]. In Marshall's paper it was proved, examining concrete program stories that, using somewhat near to Markov-property, the priority rule may reduce CPU (Central Processor Unit) time by 17%. This "Markov property" is used in our autoregressive model. The same model may be used to describe the program behaviour by working sets, proposed in the paper of Denning.

For the formal description of a computer system we use the language SIMULA. It seems to us that it is the most suitable one for our purposes. In our probabilistic model we use the whole apparatus of the theory of stochastic processes (for the inner work), where the processes depend on random vector variables. These variables have to be identified, or in mathematical statistical language, estimated from the job-flow. To illustrate our description we study special cases and give simulation results.

2. THE FORMAL DESCRIPTION

In this part we shall give for illustration a formal description of the jobs in a computer system using the language SIMULA 67. In such a way it is possible to describe, in general form, the most important characteristics of a computer system.

The reasons for the choice of the SIMULA are two: (a) The **class** concept of SIMULA is very suitable for the formal description of arbitrary structures. (b) The timing concept of a simulation language is useful to describe the timing of a computer system.

We suppose that the computer system consists of L *resources* (units). These are for example: CPU, CM-core, channels, drivers, controllers, devices, etc.

Now we introduce some notations.

Job

Definition: An L-dimensional step function $p(\tau)$ is a *job* with *execution time* T, if

*This is one of three papers appearing in this issue, which were submitted as the authorized contribution of the Hungarian Academy of Sciences to our Lánczos memorial project. (Ed.).

(a) $p(\tau) = \{p_1(\tau), \ldots, p_2(\tau)\}$, $\quad 0 \le \tau \le T$
$\quad 0 \le p_i(\tau) \le 1$

(b) $\sum^{L} p_i(\tau) > 0$ if $0 < \tau < T$ and $\sum = 0$ otherwise.

(c) $p_i(\tau) = \min\{p_i(\tau - 0), p_i(\tau + 0)\}$ for all i.

A diagram indicating this continuity concept appears in Fig. 1.

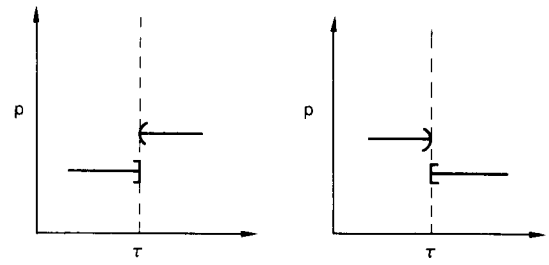

Fig. 1.

Remarks

The argument τ means the own elapsed time of the job, as it will be later exactly defined.

The function value $p_i(\tau)$ shows that at the elapsed moment τ, it must be allocated $100 \cdot p_i(\tau)\%$ of the i-th resource for this job.

Formal description

First of all we define a data structure to describe the states of the jobs:

```
class  job state;
begin  real  period;
       real  array  p[1:L];
end;
```

In this type-definition the real variable *period* means the length of the state in the elapsed time, and the vector p corresponds to the same one in the earlier definition of the job.

```
boolean  procedure  positive change (S1, S2);
                    ref  (job state) S1, S2;
             begin  boolean  b;
                    integer  j;
                    for  j:=1  step  1  until  L  do
                        b:=b  or  S1.p[j]<S2.p[j];
                    positive change:=b;
             end;
```

This procedure tests the type of a state-change. If $S2$ is the new state, and it has any allocation demand, then the value of the function becomes true.

```
process  class  job;
         virtual: ref  (job state)  procedure  next state;;
  begin  ref  (job state) present, next;
         ref  (job state)  procedure  next state;
    present: -next state;
  L: hold (present.period);
    next: -next state;
    if  next =/= none  then
          begin
            deallocation      (present, next);
```

```
            if  positive change (present, next)  then
              begin
                this  process.into (allocation  list);
                activate  (monitor); passivate;
              end;
            present: -next;
            goto  L;
        end;
    end;
```

The procedure *next state* produces a new complete data structure of type *job state*, which represents the next state of the job.

Using the SIMULA **virtual** concept, in this general form, the procedure till may be undefined.

In case of special types of jobs we can define various concrete realizations of this procedure.

For this we give an example:

```
job  class  trivial poisson job (m);  real  m;
             virtual:  ref  (job state)  procedure  next state;
          begin  integer  j;
            ref  (job state)  procedure  next state;
              begin  integer  j;
                  ref  (job state) S;
              S:-new  job state;
              S.period:=negexp (m, U);
              for  j:=1  step  1  until  L  do
                  S.p[j]:=uniform (0, 1, U);
              next state:-S;
          end;
      end;
```

Timing

Definition: We say that an indicator-function $\chi(t)$ is a *job-story* with execution time T, if

(a) $\chi(t) = \begin{cases} 1 \\ 0 \end{cases} \quad 0 \leq t \leq \infty$

(b) $\int_0^\infty \chi(t)\,dt = T$

(c) the number of jumps of function $\chi(t)$ is finite.

According to the definition, there exist such values t_{IN} and t_{TE}, for which:

$$t_{IN} = \max\{t;\ \chi(s) = 0,\ s < t\}$$
$$t_{TE} = \min\{t:\ \chi(s) = 0,\ s > t\}$$

We call t_{IN} and t_{TE} respectively by *initiating time* and *termination time* of the job-story $\chi(t)$.

Definition: If $\chi(t)$ is a job-story then we say that the function

$$\rho(t) = \int_0^t \chi(\cdot)d\cdot$$

is the *elapsed time* of $\chi(t)$. The relation between the job and job-story will be given in 4. def. 1.

Remark

If we consider the computer system as a complicated serving system, then the jobs are the customers, and suitable choosen job-stories are the realizations of the services.

Job flow

Definition: We say that the infinite set

$$\{\mathbf{J}_j\} = \{t_j, \mathbf{p}^{(j)}(\tau)\} \quad j = 1, 2, 3, \ldots$$

is a *job-flow* or job-stream, if $\mathbf{p}^{(j)}(\tau)$ are jobs, and $t_j \geq 0$, $t_{j+1} \geq t_j$.

Here t_j means the arrival time of the $\mathbf{p}^{(j)}(\tau)$ job. From this definition it follows: If a job-story $\chi^{(j)}(t)$ is the realization of the job $\mathbf{p}^{(j)}(\tau)$ then $t_{IN}^{(j)} \geq t_j$.

3. A GENERAL PROBABILISTIC MODEL

In our conception the $p_i(\tau)$ functions are random step functions, which may be approximated in some cases by continuous functions. This stochastic model is a good description in spite of the fact that, for a given job, the whole function is observable. This random function (or stochastic process) principle may be used successfully when applied to multiprogrammed computer descriptions which deal with longer periods of time (e.g. for several hours or for a day).

The *inner probability field* is used to describe the changes of the functions $p_i(\tau)$, $i = 1, 2, \ldots L$ in the inner time of the computer. Naturally, this probability field depends upon the type of problem we want to solve. Initially, it was supposed that the statistical behaviour of these processes could be described by queueing processes, Markov and semi-Markov processes. Here we go a bit further; we apply diffusion-type approximations.

In addition to the inner probability field, it seems natural to introduce an *outer probability field* which describes the job population. In most cases we assume that this is a multidimensional stationary process. This field is related to job-flow $\{\mathbf{J}_i\}$ and also to the t_j time points, while the inner-field to the jobs $\boldsymbol{p}(\tau)$ and also to elapsed time τ, as was defined in the proceeding paragraph. The whole description of a computer system with a job-flow is given as the product of the initial probability field, the outer-field, and the conditional inner field which is determined by a given job. The idea is the same as in the case of a Brownian motion process with initial distribution.

Let us see an example. We assume that $\{t_i\}$ is a Poisson process, and further

$$\boldsymbol{p}(\tau) = \{p_1(\tau), p_2(\tau)\}$$

where $0 \leq p_1(\tau) \leq 1$ denotes the CM (Central Memory)—core demand ($p_1 = 1$ when the whole available memory is demanded), p_2 is the CPU-indicator function, i.e.

$$p_2(\tau) = \begin{cases} 1 & \text{if the job demands the CPU} \\ 0 & \text{if not.} \end{cases}$$

τ denotes the elapsed time of the job.

The following properties we assume

$$p_1(\tau) = \xi_{11} g_1(\tau, \xi_{12})$$

where the parameters ξ_{11}, ξ_{12} are constants for the job, but they are random variables on the outer probability field, as they depend on the job. ξ_{11} denotes the maximal core-demand of the job, while ξ_{11} denotes the maximal core-demand of the job, while ξ_{12} characterizes the density of changing. The function $g_1(\tau, \xi_{12})$ is a random step function, $0 \leq g_1 \leq 1$ where the length of intervals of constant function values are exponentially distributed with parameter ξ_{12}. To describe $p_2(\tau)$ in the same way we use the following "integral" connection:

$$\tilde{p}_2(kS) = \int_{(k-1)S}^{kS} p_2(\tau)\, \mathrm{d}\tau$$

where S is so large that $\tilde{p}_2$ is nearly normally distributed and in consecutive S long time intervals $p_2^* = \tilde{p}_2 - E\tilde{p}_2$ fullfils the difference equation

$$p_2^*(kS) + \xi_{21} p_2^*((k-1)S) + \xi_{22} p_2^*((k-2)S) = \epsilon_2(k)$$

where $\epsilon_2(k)$ is an independent Gaussian sequence with mean 0 and variance σ_ϵ^2.

This means, that the sequence $p_2(kS)$ is stationary and has a period if the equation

$$z^2 + \xi_{21} z + \xi_{22} = 0$$

has complex roots. The parameters ξ_{21}, ξ_{22} are again constant for the given job, but they are random variables on the outer probability field. This autoregressive "periodicity" in the jobs, in quite another way, we may find in Denning's work[7].

As the random vector $\{\xi_{21}, \xi_{22}, \sigma_2^2\}$ is defined on the outer probability field, we can identify it if the distribution is now known.

In the general case we suppose that

$$p_i(\tau) = g_i(\tau, \underline{\xi}_i)$$

where $\underline{\xi}_i$ is a random vector on the external probability field, and for given $\underline{\xi}$; $g_i(\tau, \underline{\xi}_i)$ is a stochastic process on the inner field. In many cases g_i has the form

$$g_i(\tau, \underline{\xi}_i) = \sum_{k=1}^{N} \xi_{ik} h_k(\tau, \xi'_{ik})$$

where N is fixed. In this case we have regression type problems when the distributions of ξ_{ik} are unknown.

Other approximations the reader may find in works Arató–Töke–Knúth[1], Arató[2].

As an example of the description of the outer probability field we give the main statistical characteristics of debugging period of our CDC 3300 computer.

The average CM-core demand is 65% (dispersion 21%). The average SCR-file demand is 15% (dispersion 12%). The average total CPU time is 50 sec (dispersion 93). The average total channel time is 22 sec (dispersion 39).

Using cluster analysis technics we got six significant clusters:

1. Compilation (26%).
2. Small jobs (13%).
3. Medium jobs (12%).
4. Large CPU, large SCR (14%).
5. Large CPU, small SCR (15%).
6. Long jobs (14%).

Non clustered 6%.

4. JOB CONTROL

Job-phases

Definition: Let the job-story $\chi(t)$ be the realization of the job $p(\tau)$. We say that at the moment t, p is in *active-phasis* if $\chi(t) = 1$ and in *waiting-phasis* if $\chi(t) = 0$.

Remarks

If a state of a job is finished, and the change is positive, the job alarms the *monitor*, and hands over the demanded $p(\tau + 0)$ state to it. Than, depending on the free capacity of the resources and on the scheduling strategies of the monitor, the job has to *wait* or not.

If the monitor is alarmed then there are three possibilities:

(a) Allocation is possible and allowed by the monitor.

(b) Allocation is possible, but the monitor protects the resource for another demand.

(c) Allocation is impossible.

Waiting-phases

(1) Until the reactivation of the job the elapsed time of the job remains constant.

(2) The allocated resources for the job during the waiting time is $p(\tau_0)$, where τ_0 is the (elapsed) moment when the positive state change occurred. According our continuity concept, new resources are yet not allocated, but the unsufficient resources are released.

Forced-state

(a) The monitor can interrupt and force to wait a job without any change of its natural state.

(b) The monitor can force a waiting job to release some of its allocated resources.

It is easy to see that such manipulations are equivalent to the modification of the arrived jobs only in finitely many points.

Of course the execution of such forcing algorithms causes additional costs. Such an operating system algorithm may be very useful, if it generally ensures extra utilization, and the forced cases occur only with very small probability. Such an example was solved in the diploma of Kovács [8].

To solve the widely known deadlock problem of resource allocation also forcing algorithms are used. See for example Coffman [6].

Monitor strategy

Definition: A *strategy* is a mapping

$$\chi = S(\boldsymbol{J})$$

which results a job-story χ for all elements $\boldsymbol{J} = \{t, \boldsymbol{p}\}$ of a job-stream, and fulfills the following conditions. Let $\boldsymbol{J}_j$ be the j-th element of the job-stream $\boldsymbol{J}$ and denote

$$\chi_j(t) = S(\boldsymbol{J}_j) = S(t_j, \boldsymbol{p}^{(j)}(\tau))$$

Let be $t_{IN}^{(j)}$, $t_{TE}^{(j)}$, $T^{(j)}$ the initiating-, termination-, and execution times of χ_j, and T_j the execution time of $\boldsymbol{p}^{(j)}$.

Using this notation the following conditions must hold:

(1) $T^{(j)} = T_j$

(2) $t_{IN}^{(j)} \geq t_j$

(3) $\sum_j p_i^{(j)}(\rho_j(t)) \geq 1, \quad \forall i, t.$

where ρ_j is the elapsed time corresponding to χ_j.

(4) $\sum_j \chi_j(t) \geq 1, \quad \forall t.$

Remarks

The first two conditions are trivial. The third condition means that no one of the resources can be allocated in more than 100%. The fourth condition means that the system never can be deadlocked.

System state

Definition: If S is a monitor strategy, then the *system-state* is a vector $\boldsymbol{q}(t) = \{q_1(t), \ldots, q_2(t)\}$ defined by

$$q_i(t) = \sum_j p_i^{(j)}(\rho_j(t)).$$

Remarks

As $p_i(\tau) = 0$ if $\tau = 0$ or $\tau = T$, the above Σ consists only of the allocated demands of the submitted, but not terminated jobs.

If S is a forcing strategy the definition remains correct too, because S is forcing to decrease some $p_i(\tau)$ demands.

We can write the third condition in the definition of the strategie as $q_i(t) \leq 1$ for all i, t.

Monitor-state

The monitor, as an infinite looping job also, has its own allocation demands. Let us denote the state vector $\boldsymbol{p}(\tau)$ of the monitor by $\boldsymbol{p}^{(0)}(t)$

$$\boldsymbol{p}^{(0)}(t) = \{p_1^{(0)}(t), \ldots, p_L^{(0)}(t)\}.$$

The $p^{(0)}(t)$ vector, as a stochastic process is defined on the earlier mentioned two probability

fields. It is determined by the whole job-flow, its realizations and by the monitor strategy. The last means a stochastic control, which is not a linear one.

The monitor-state changes if some well defined situations occur handled by S. Such monitor activities are for example: suspension, paging, etc.

Formal description

```
process class monitor;
 begin ref (job state) monitor state;
       ref (job) J;
     monitor state:— idle state;
 idle: passivate;
 alarm: for J:—allocation list.first, J.suc while
    begin                                    [J =/= none do]
      if allowed (J) then begin
                 allocation;
                 reactivate J;
                 end else
      if forcing state (J) then begin

                 change monitor state;
                 force states;
                 hold (monitor action);
                 monitor state:— idle state;
                 reactivate J;
                 end
        else J. wait (allocation list);
        end;
        goto idle;
    end;
```

Maximal strategy

Definition: The strategy S is *maximal* if for any job J_m of waiting phasis

$$q_i(t) + p_i^{(m)}(\rho_m(t) + 0) > 1$$

for all $t > 0$ and for at least one $i = \{1, \ldots, L\}$.

Remark

If a strategy is not maximal we can improve the utilization at least in one resource by activation of a waiting job.

Performance measures of strategies

(1) The multiprogramming level is defined as

$$\eta_m = \lim_{v \to \infty} \frac{1}{v} \int_0^v k_M(t)\,\mathrm{d}t$$

if it exists, where $k_M(t)$ is the number of submitted, but not terminated jobs at the moment t.

(2) The activization level is defined as

$$\eta_A = \lim_{v \to \infty} \frac{1}{v} \int_0^v k_A(t)\,\mathrm{d}t$$

if it exists, where $k_A(t)$ is the number of jobs, for which $\chi_i(t) = 1$.

(3) The definition of utilization of the k-th resource:

$$\nu_k = \lim_{v \to \infty} \frac{1}{v} \int_0^v q_k(t)\,dt.$$

The above limits exist if the processes $k_M(t)$, $k_A(t)$, $q_k(t)$ are stationary. This depends not only on the job-stream, but on the monitor-strategy too. This is a general problem of our non-linear stochastic control.

In the case of finite job-streams "performance test sets" we can state some simple facts: Since

$$\int_0^T k_A(t)\,dt = \sum_{j=1}^{N} T^{(j)}$$

where N is the number of jobs in the set, and T is the total execution time of the set, therefore

$$\eta_A = \frac{\Sigma T^{(j)}}{T} = \frac{\text{const}}{T}.$$

Hence the total execution time T is minimal if the activation level is maximal.

This fact is quite natural, but a similar statement for the multiprogramming level does not hold.

On the other hand

$$\int_0^T q_k(t) = \sum_j^N \int_0^{T^{(j)}} p_k^{(j)}(t) = C \text{ (const)}.$$

and this means

$$\nu_k = \frac{C}{T}.$$

Hence we obtain that the maximality of the utilization is equivalent for all the resources.

5. SPECIAL CASES

There are such resources which we can allocate only exclusively. Such resources we define as *singular units*, while the others *regular units*.

Between the components of the jobs which respect to singular units there are tight connections. At the same time only well defined groups can take value 1. These values can remain constant during a sequence of job-states.

Taking into account the change of singular components we can define *main states* of the jobs.

In the special case, when always exactly one of the singular components has value 1, while the others all have zero we can identify the main states with the name of the used singular unit.

A further specialization is when only two main states may be, namely CPU-period and IO-period. Most of the results are known in this case, see Tomkó[3], Gaver[5], Denning[7].

The regular units are often handled statically only. In this case

$$p_i(t) = \xi f(t, \eta)$$

where the random variable ξ, η are defined on the outer probability field, and $f(t, \eta)$ is a random indicator function of a time interval, defined on the inner probability field.

The CPU scheduling problem

One of the most important activities of the monitor is the optimal scheduling of the CPU. The difficulty of the problem is that to find the optimal strategy, which ensures maximal CPU utilization. leads to a nonlinear stochastic control problem, in the simplest cases too. If the components, $p_i^{(j)}(\tau)$, $i = 1, \ldots, ; j = 1, 2, \ldots$ of the jobs are stationary processes, then because of the monitor strategy, the resulting component-processes of the system state may well be non-stationary.

The simplest case of CPU scheduling is the following:

$L = 2$, and the two resource is two singular units: the CPU and an I/O unit. The jobs have two states (main states) CPU and IO periods. This problem is studied under different conditions on the distributions of the length of the periods, by Gaver[5], Tomkó[3], Arató[2], and using simulation methods by Marshall[4].

Because of the latent periodicity of the changes of the length of these periods in the jobs it seems natural to use second order autoregresive processes to describe the probabilistic behaviour of the $p_1(\tau)$, $p_2(\tau)$ functions. This case is studied by Arató–Knúth–Töke[1].

Finally we give some simulation results for this case. We assume that the length of the CPU and IO periods are exponentially distributed with parameters $\lambda(t)$ and $\mu(t)$ respectively, and

$$\lambda(t) = \frac{1}{E} + \lambda'(t)$$

$$\mu(t) = \frac{1}{I} + \mu'(t)$$

where $\lambda'(t)$ and $\mu'(t)$ are second order autoregressive processes, and E, I are constants.

Let ρ be the traffic intensity:

$$\rho = \sum \frac{E}{E+I}$$

Consider the following CPU scheduling strategies:

1. FIFO
2. scheduling the maximal $I(t)/(E(t)+I(t))$
3. scheduling the minimal $I(t)/(E(t)+I(t))$ where $E(t)$ and $I(t)$ means the expected length of CPU and I/O intervals at the moment t.

Then, for 3 jobs the CPU idle time in percent is the following:

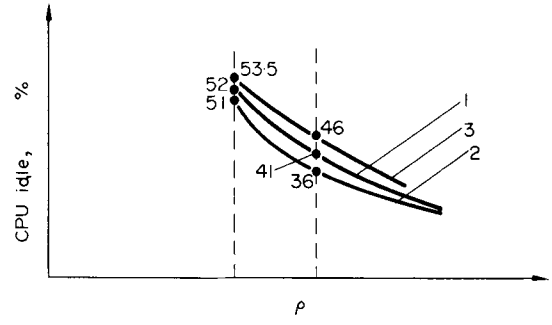

Fig. 2.

And the same for 8 jobs:

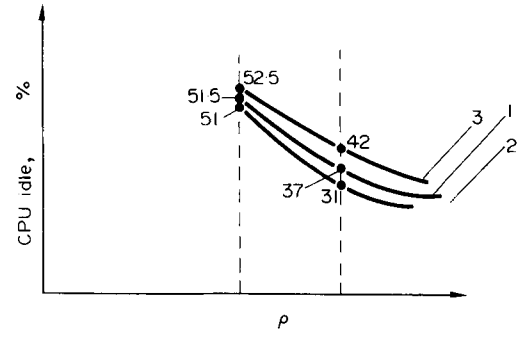

Fig. 3.

For more detailed description, and results of the simulation are in [10].

Acknowledgements—The author is very grateful for the helpful suggestions of the members of our working group in the Computer and Automation Institute of Hungarian Academy of Sciences, mainly to Prof. M. Arató, Dr. J. Tomkó, and P. Töke.

REFERENCES

1. M. Arató, E. Knuth and P. Töke, On stochastic control of a multiprogrammed computer based on a probabilistic model. *IFAC Symposium on Stochastic Control*, Budapest, 305–311 (1974).
2. M. Arató, Diffusion approximation for multiprogrammed computer systems. *Comp. & Maths. with Appls.* **1**, 315–326 (1975).
3. J. Tomkó, Processor utilization study. *Comp. & Maths. with Appls.* **1**, 337–344 (1975).
4. B. S. Marshall, Dynamic calculation of dispatching priorities under OS/360 MVT, *Datamation* **9**, 93–97 (1969).
5. D. P. Gaver, Probability models for multiprogramming computer systems, *J. ACM.* **14**, 423–438 (1967).
6. E. G. Coffman, Deadlocks in computer systems, Operating systems, Infotech State of the Arts, Report, 353–358 (1972).
7. P. Denning, On modeling program behaviour, *Proceedings AFIPS Conf.* **40**, 937–944 (1972).
8. K. Kovács, Analysis of the memory oversubscription of computers using Markov processes, (in Hungarian), Dissertation, University of Sciences, Budapest (1974).
9. E. Knuth, Analysis of input strategies of multiprogramming computer systems, (in Hungarian). *Proceedings of Conference on Computer Science*, Hungary, Székesfehérvár, 59–62 (1973).
10. E. Knuth, Simulation study of CPU utilization using autoregressive processes. *Commun. Comp. Automation Inst. Hung. Acad. Sci.* (1975).
11. E. Knuth, Multiprogramming computer systems, (in Hungarian). *Mérés Autom.* **5**, 208–211 (1974).